Comp. & Maths. with Appls., Vol. 5, pp. 59–73
Pergamon Press Ltd., 1979. Printed in Great Britain

0097–4943/79/0301–0059/$02.00/0

ACCELERATION TECHNIQUES FOR A CLASS OF x-PROJECTION METHODS FOR SOLVING SYSTEMS OF LINEAR EQUATIONS

ROGER L. WAINWRIGHT
The University of Tulsa, Tulsa, OK 74104, U.S.A.

Communicated by Ervin Y. Rodin

(*Received May* 1978; *in revised form September* 1978)

Abstract—The solution of linear systems of equations using a 2-dimensional x-projection method is presented. At each step of the iterative process the approximate solution vector is projected to a point in the intersection of two of the hyperplanes of the linear system. It is shown that nonsingularity of the coefficient matrix is the only requirement for convergence. An algorithm is presented to select pairs of hyperplanes to project the approximate solution vector at each step. The algorithm is quasi-optimal since the hyperplanes, which are determined by the row vectors of the coefficient matrix, are selected *a priori*. This is shown to significantly reduce the number of cycles required for convergence. We observe that in some cases the ratio of the change vectors of the approximate solution vectors after some number of cycles becomes a constant. Thus, when this occurs a simple geometric acceleration can be applied to calculate the solution directly. This is shown to significantly reduce the number of cycles required for convergence and improves the accuracy of the solution by orders of magnitude. The 2-dimensional x-projection method was tested against other well known iterative algorithms over a wide range of linear systems and proved superior (less C.P.U. time required) in nearly every case.

1. INTRODUCTION

The purpose of this paper is to develop a class of projection methods called x-projection methods for solving the equation

$$Ax = b \tag{1.1}$$

when A is a nonsingular matrix of order n and x and b are n-vectors. In this paper we also develop some acceleration techniques for the class of x-projection methods. The paper is divided into six sections. In Section 2 the notion of projection methods for solving systems of linear equations is reviewed and two classes of projection methods called r-projection and x-projection methods are defined. In Section 3 a 2-dimensional x-projection method is developed and in Section 4 a hyperplane selection criteria is developed to accelerate the rate of convergence of the 2-dimensional x-projection method. The major advantage of the x-projection methods over the traditional r-projection methods is that the ratio of the changes to the approximate solution vectors from cycle to cycle become a scalar multiple of each other. Hence, after some number of iterations the process can be stopped and the solution calculated directly. This notion is also presented in Section 4. Test cases and comparisons are presented in Section 5. Concluding remarks are presented in Section 6 followed by a program listing of the 2-dimensional x-projection method using a quasi-optimal hyperplane selection criteria and geometric acceleration.

2. BASIC CONCEPTS OF PROJECTION METHODS

Consider the following non-stationary iterative scheme for solving (1.1) that makes successive changes to the components of a starting approximation vector, x^0

$$x^{k+1} = x^k + \alpha_k \omega^k \tag{2.1}$$

where α_k is a scalar and ω^k is an n-vector. Various algorithms come from different choices of α_k and ω^k. Define the error vector, r^k after the kth step of (2.1) as

$$r^k = b - Ax^k.$$

Consider the first possibility of choosing α_k and ω^k to force the residual vector element, r_i^k, to zero at each iterative step. This gives rise to a class of methods called relaxation methods. Let $\alpha^k = c(r_i^k/(a_i, a_i))$ and $\omega^k = e^i$ where (a_i, a_i) is the inner product of the ith column of A and e^i is the ith column of the unit matrix, then (2.1) becomes

$$x^{k+1} = x^k + c\frac{r_i^k}{(a_i, a_i)}e^i \tag{2.2}$$

where c is the relaxation parameter. For $c = 1$ (2.2) yields the method of Gauss–Seidel and for $c \neq 1$ we have an over relaxation or under relaxation method.

A. r-*projection methods*

Consider a class of methods called gradient methods which minimizes at each step of (2.1), the following quadratic form

$$(r^k, r^k) = (b - Ax^k, b - Ax^k). \tag{2.3}$$

For the most substantial reduction of (2.3), that is to minimize (r^k, r^k) with respect to α_k we choose

$$\alpha_k = \frac{(A\omega^k, r^k)}{(A\omega^k, A\omega^k)} \tag{2.4}$$

and ω^k is as yet unspecified. Now if ω^k is chosen as the vector of steepest descent i.e.,

$$\omega_i^k = \frac{\partial(r^{k+1}, r^{k+1})}{\partial x_i^k}$$

then (2.1) becomes the method of steepest descent or sometimes called the gradient method. If however, ω^k is choosen as e^i then (2.1) becomes

$$x^{k+1} = x^k + \frac{(r^k, a_i)}{(a_i, a_i)}e^i$$

which changes just one component, (the ith component) at each step of the iterative scheme. Geometrically, at each step of the iterative process the residual vector, r^k, is projected onto a_i and the new residual, r^{k+1}, is orthogonal to a_i. We call this a 1-dimensional r-projection method since r^k is being projected onto one of the column vectors of A. This method was initially developed by A. de la Garza[2] and rediscovered by Keller[6]. Fox[1, 205–206] also gives a development of this method.

Let $\omega^k = e_1 + e_2 + \cdots e_m$ then an extention of the general scheme (2.1) to m-dimensions yields the m-dimensional r-projection method

$$x^{k+1} = x^k + d_1{}^k e_1 + d_2{}^k e_2 \cdots + d_m{}^k e_m$$

where $x_1, x_2, x_3, \ldots, x_m$ are m arbitrary components of the approximate solution vector. d_i^k is the change to the ith component of x^k at the kth iteration. Geometrically, at each step r^k is projected onto a subspace determined by m of the column vectors of A and the new residual, r^{k+1}, is orthogonal to this subspace. This is geometrically described by Householder[5]. The changes to m components of the approximate solution vector at the kth step of an m-dimensional r-projection method are given as the solution to the following symmetric system of m linear equations

$$\begin{aligned}
&a_{11}d_1{}^k + a_{12}d_2{}^k + a_{13}d_3{}^k \cdots + a_{1,m}d_m{}^k = (r^k, a_1)\\
&a_{21}d_1{}^k + a_{22}d_2{}^k + a_{23}d_3{}^k \cdots + a_{2,m}d_m{}^k = (r^k, a_2)\\
&\vdots\\
&a_{m,1}d_1{}^k + a_{m,2}d_2{}^k + a_{m,3}d_3{}^k \cdots + a_{m,m}d_m{}^k = (r^k, a_m)
\end{aligned}$$

where $a_{ij} = (a_i, a_j)$. A proof of this can be found in Wainwright[10]. The most significant characteristic of the r-projection class of methods is that nonsingularity of the coefficient matrix is sufficient for convergence[6]. The rate of convergence, however, is slow and various techniques for increasing the rate of convergence can be found in [3, 8, 9, 11].

B. x-*projection methods*

Pizer[7, 162–168] gives a detailed development with geometrical interpretation and program listing for what we call the 1-dimensional x-projection method. He states this method was discovered by Kaczmarz and rediscovered by Levy. Briefly the method is as follows:

Each equation of (1.1) (i.e. $(a^i, x) = b_i$ where a^i is the ith row of A) geometrically defines an $(n-1)$-dimensional hyperplane called the ith hyperplane which is orthogonal to a^i. The solution vector is the point in which the n hyperplanes of (1.1) intersect. The iterative process begins with an arbitrary point, x^0, then moves it onto hyperplane 1 in a direction orthogonal to it. It then moves to hyperplane 2 in a direction orthogonal to it, and so on to hyperplane n in a direction orthogonal to it. Then the process starts over by moving to hyperplane 1 in a direction orthogonal to it, to hyperplane 2 in a direction orthogonal to it and so on. Pizer proves that this process always converges for arbitrary x^0 assuming the nonsingularity of A. He shows when we move to hyperplane $i+1$ from hyperplane i (solve the $(i+1)$th equation) we are moving closer to the intersection of all of the hyperplanes (the solution). Algebraically the projection step of moving from the ith hyperplane to the $(i+1)$th hyperplane is written

$$x^{i+1} = x^i + \frac{(b_{i+1} - (a^{i+1}, x^i))}{(a^{i+1}, a^{i+1})} a^{i+1}.$$

To save time in the iteration we scale each equation according to the Euclidean norm so that $(a^{i+1}, a^{i+1}) = 1$, then the projection step becomes

$$x^{i+1} = x^i + (b_{i+1} - (a^{i+1}, x^i))a^{i+1}. \tag{2.5}$$

This is a total step method in that every component of the approximate solution vector is altered at each step. If we apply (2.5) successively for $i = 0, 1, 2, \ldots, n-1$ the projections constitute a full cycle. So that we can represent successive cycles we define x^{mn+i+1} as the result of the projection to hyperplane $i+1$ from hyperplane i in the $(m+1)$th cycle ($m = 0$ initially). Now the projection step (2.5) can be rewritten as

$$x^{k+1} = x^k + (b_i - (a^i, x^k))a^i$$

or equivalently

$$x^{k+1} = x^k + r_i^k a^i \tag{2.6}$$

where i = the residue of $k+1$ modulo n. Thus in the general scheme (2.1) the 1-dimensional x-projection method, (2.6), is obtained when $\alpha_k = r_i^k$ and $\omega^k = a^i$.

3. 2-DIMENSIONAL x-PROJECTION METHOD

In the previous section we developed from (2.1) a 1-dimensional r-projection method (residual vector projected onto one of the columns of A) and the 1-dimensional x-projection method (the approximate solution vector projected onto one $(n-1)$-dimensional hyperplane). Considerable work has been done with higher dimensional r-projection methods and they have been shown to be competitive to other well known iterative methods[3, 8–11]. To the author's knowledge higher dimensional x-projection methods have not been explored. In this section the 2-dimensional x-projection method is developed. Higher dimensional x-projection methods can be easily developed in a similar manner.

Consider the following non-stationary iterative scheme for solving (1.1) that makes successive changes to the components of a starting vector, x^0

$$x^{k+1} = x^k + \alpha_k \omega^k + \beta_k \gamma^k$$

where α_k and β_k are scalars and ω^k and γ^k are n-vectors. This is an extension of the general scheme (2.1). For the 2-dimensional x-projection method the iterative process begins with an arbitrary point, x^0, then moves it onto the intersection of hyperplanes 1 and 2 in a direction orthogonal to both of them. It then moves to the intersection of hyperplanes 3 and 4 in a direction orthogonal to both of them, and so on to the intersection of hyperplanes $n-1$ and n in a direction orthogonal to both of them. Then the process starts over by moving to the intersection of hyperplanes 1 and 2 in a direction orthogonal to both of them, then to hyperplanes 3 and 4 in a direction orthogonal to both of them and so on. More precisely, if n is even the hyperplanes are paired $(1,2), (3,4), \ldots, (n-1, n)$ and if n is odd the hyperplanes are paired $(1,2), (3,4), \ldots, (n-2, n-1), (n-1, n)$. Let x^k be a vector to a point in the intersection of the ith and $(i+1)$th hyperplanes. Let x^{k+1} be the vector to the point in the intersection of the $(i+2)$th and $(i+3)$th hyperplanes which is reached by moving from x^k in a direction orthogonal to both the $(i+2)$th and $(i+3)$th hyperplanes. Let y be any vector to the point in the intersection of hyperplanes $i+2$ and $i+3 (y \neq x^{k+1})$. Then the change in the approximate solution vector, $x^{k+1}-x^k$, will be orthogonal to $x^{k+1}-y$, which lies in the intersection of the $(i+2)$th and $(i+3)$th hyperplanes:

$$((x^{k+1}-x^k), (x^{k+1}-y)) = 0 \tag{3.1}$$

Since both x^{k+1} and y define points in the intersection of the $(i+2)$th and $(i+3)$th hyperplanes, both satisfy the $(i+2)$th and $(i+3)$th equations of (1.1):

$$\begin{aligned} (a^{i+2}, x^{k+1}) &= b_{i+2} \\ (a^{i+2}, y) &= b_{i+2} \end{aligned} \tag{3.2}$$

$$\begin{aligned} (a^{i+3}, x^{k+1}) &= b_{i+3} \\ (a^{i+3}, y) &= b_{i+3} \end{aligned} \tag{3.3}$$

thus

$$(a^{i+2}, (x^{k+1}-y)) = 0 \tag{3.4}$$

and

$$(a^{i+3}, (x^{k+1}-y)) = 0. \tag{3.5}$$

Therefore, a linear combination of a^{i+2} and a^{i+3} will be orthogonal to $x^{k+1}-y$ for any y to the point in the intersection of hyperplanes $i+2$ and $i+3$ (y can range over $n-2$ dimensions). Thus it follows from (3.1), (3.4) and (3.5) that a linear combination of a^{i+2} and a^{i+3} will be in the same direction as $x^{k+1}-x^k$, so, for scalars α and β,

$$x^{k+1} = x^k + \alpha a^{i+2} + \beta a^{i+3}. \tag{3.6}$$

By taking the inner products of a^{i+2} with (3.6) and a^{i+3} with (3.6) producing

$$(a^{i+2}, x^{k+1}) = (a^{i+2}, (x^k + \alpha a^{i+2} + \beta a^{i+3}))$$

and

$$(a^{i+3}, x^{k+1}) = (a^{i+3}, (x^k + \alpha a^{i+2} + \beta a^{i+3}))$$

respectively, and then using (3.2) and (3.3) we obtain the following symmetric linear system of two equations and two unknowns:

$$\begin{aligned} \alpha(a^{i+2}, a^{i+2}) + \beta(a^{i+2}, a^{i+3}) &= b_{i+2} - (a^{i+2}, x^k) \\ \alpha(a^{i+3}, a^{i+2}) + \beta(a^{i+3}, a^{i+3}) &= b_{i+3} - (a^{i+3}, x^k). \end{aligned} \tag{3.7}$$

Solving (3.7) for α and β we get

$$\alpha = \frac{[(a^{i+3}, x^k) - b_{i+3}](a^{i+2}, a^{i+3}) - [(a^{i+2}, x^k) - b_{i+2}]}{1 - (a^{i+2}, a^{i+3})^2}$$
$$\beta = \frac{[(a^{i+2}, x^k) - b_{i+2}](a^{i+2}, a^{i+3}) - [(a^{i+3}, x^k) - b_{i+3}]}{1 - (a^{i+2}, a^{i+3})^2}. \tag{3.8}$$

Thus the 2-dimensional x-projection method is defined by (3.6) where α and β are defined in (3.8).

THEOREM 3.1. A single step of a 2-dimensional x-projection method projecting x^k, which defines a point in the intersection of hyperplanes $i-2$ and $i-1$, to x^{k+1} which defines a point in the intersection of hyperplanes i and $i+1$ in a direction orthogonal to hyperplanes i and $i+1$, is equivalent to repeated 1-dimensional x-projection steps projecting the approximate solution vector back and forth to points in the ith and $(i+1)$th hyperplanes until the change between two successive approximate solution vectors is zero.

Proof. Given x^k in a 1-dimensional x-projection scheme we generate x^{k+1} as a vector to a point in the ith hyperplane by the following 1-dimensional x-projection scheme

$$x^{k+1} = x^k + c_{ik}a^i \tag{3.9}$$

where

$$c_{ik} = b_i - (a^i, x^k). \tag{3.10}$$

Next x^{k+2} is generated as a vector to a point in the $(i+1)$th hyperplane by the same process producing

$$x^{k+2} = x^{k+1} + c_{i+1,k+1}a^{i+1}. \tag{3.11}$$

In general

$$c_{i+2j,k+2j} = c_{i,k+2j} = b_i - (a^i, x^{k+2j})$$

and

$$c_{i+1+2j,k+1+2j} = c_{i+1,k+1+2j} = b_{i+1} - (a^{i+1}, x^{k+1+2j})$$

for $j = 0, 1, 2, 3, \ldots$.

Substituting (3.9) in (3.11) we obtain

$$x^{k+2} = x^k + c_{ik}a^i + c_{i+1,k}a^{i+1} - c_{ik}(a^i, a^{i+1})a^{i+1}. \tag{3.12}$$

Continuing, x^{k+3} is generated as a vector to a point in the ith hyperplane and x^{k+4} as a vector to a point in the $(i+1)$th hyperplane:

$$x^{k+3} = x^{k+2} + c_{i,k+2}a^i \tag{3.13}$$

$$x^{k+4} = x^{k+3} + c_{i+1,k+3}a^{i+1}. \tag{3.14}$$

Substituting (3.12) and (3.13) in (3.14), it can be rewritten in terms of x^k.

$$\begin{aligned} x^{k+4} = x^k &+ [c_{ik} - c_{i+1,k}(a^i, a^{i+1}) + c_{ik}(a^i, a^{i+1})^2]a^i \\ &+ [c_{i+1,k} - c_{ik}(a^i, a^{i+1}) + c_{i+1,k}(a^i, a^{i+1})^2 - c_{ik}(a^i, a^{i+1})^3]a^{i+1}. \end{aligned} \tag{3.15}$$

Continuing in this recurrsive manner by projecting back and forth between hyperplanes i and $i+1$ an infinite number of times we produce

$$\begin{aligned} \lim_{t\to\infty} x^{k+t} = x^k &+ [c_{ik} + c_{ik}(a^i, a^{i+1})^2 + c_{ik}(a^i, a^{i+1})^4 + c_{ik}(a^i, a^{i+1})^6 + \cdots]a^i \\ &- [c_{i+1,k}(a^i, a^{i+1}) + c_{i+1,k}(a^i, a^{i+1})^3 + c_{i+1,k}(a^i, a^{i+1})^5 + \cdots]a^i \\ &+ [c_{i+1,k} + c_{i+1,k}(a^i, a^{i+1})^2 + c_{i+1,k}(a^i, a^{i+1})^4 + c_{i+1,k}(a^i, a^{i+1})^6 + \cdots]a^{i+1} \\ &- [c_{ik}(a^i, a^{i+1}) + c_{ik}(a^i, a^{i+1})^3 + c_{ik}(a^i, a^{i+1})^5 + \cdots]a^{i+1}. \end{aligned} \tag{3.16}$$

We scale each equation according to the Euclidean norm so that $(a^i, a^i) = 1$. Using the Cauchy–Schwarz inequality we know

$$|(a^i, a^{i+1})|^2 \leq (a^i, a^i)(a^{i+1}, a^{i+1}) = 1.$$

In fact $|(a^i, a^{i+1})|^2 = 1$ only when $a^{i+1} = a^i$ which is impossible for a nonsingular coefficient matrix; thus $|(a^i, a^{i+1})|^2 < 1$. We can now apply the geometric series

$$1 + q + q^2 + q^3 + \cdots = \frac{1}{1-q}, \quad -1 < q < 1$$

where $q = (a^i, a^{i+1})^2$ to (3.16):

$$\lim_{t\to\infty} x^{k+t} = x^k + \frac{c_{ik} - c_{i+1,k}(a^i, a^{i+1})}{1-(a^i, a^{i+1})^2}\, a^i + \frac{c_{i+1,k} - c_{ik}(a^i, a^{i+1})}{1-(a^i, a^{i+1})^2} a^{i+1}. \tag{3.17}$$

Finally, substituting for c_{ik} and $c_{i+1,k}$ using (3.10), (3.17) becomes identically the equation for the 2-dimensional x-projection scheme as defined in (3.6) and (3.8). This is the desired result.

THEOREM 3.2. The 2-dimensional x-projection method for solving linear systems is convergent provided the coefficient matrix is nonsingular.

Proof. The proof follows immediately from theorem 3.1. The 2-dimensional x-projection method is based on repeated applications of the 1-dimensional x-projection method which is always convergent. The Pythagorean theorem guarantees that we move closer to the intersection of two hyperplanes as we project back and forth between them in directions orthogonal to them. In fact it can be shown that we move closer to the intersection of all of the hyperplanes [7].

4. ACCELERATION TECHNIQUES FOR THE 2-DIMENSIONAL x-PROJECTION METHOD

A. *Selection of projection hyperplanes*

The x-projection class of methods do not minimize (r^k, r^k) at each step. Thus at each step the approximate solution vector is projected closer to the intersection of all of the hyperplanes defined by the linear system but does guarantee to decrease the length of the residual vector. It is for this reason that for some systems the 1-dimensional x-projection method converges faster than the 2-dimensional x-projection and for other systems the reverse is true. It has been the author's experience, however, for random sequences of hyperplanes that the 2-dimensional x-projection method in general converges slightly faster (less C.P.U. time) than the 1-dimensional x-projection method.

The x-projection class of methods is typical of most projection methods in that the rate of convergence is very slow. One would like to be able to select *a priori*, a sequence of pairs of projection hyperplanes which guarantees most rapid convergence for a system of linear equations. One criterion for selecting an optimal sequence of projections is to observe how the residual at any step is related to the hyperplanes determined by two of the rows of A. In particular one would like to reduce the length of the residual vector as much as possible. Let x^k be a vector to a point in the intersection of hyperplanes p and q and we move x^k to x^{k+1} along a direction orthogonal to hyperplanes i and j using a 2-dimensional x-projection method. Thus

$$x^{k+1} = x^k + \alpha a^i + \beta a^j$$

and

$$(r^{k+1}, r^{k+1}) = (b - Ax^{k+1}, b - Ax^{k+1})$$

or equivalently

$$(r^{k+1}, r^{k+1}) = (r^k, r^k) - 2\alpha(r^k, Aa^i) - 2\beta(r^k, Aa^j) + \alpha^2(Aa^i, Aa^i) + 2\alpha\beta(Aa^i, Aa^j) + \beta^2(Aa^j, Aa^j). \tag{4.1}$$

Substituting (3.8) for α and β and $\cos\theta_{ij}$ for (a^i, a^j) (since $(a^i, a^i) = (a^j, a^j) = 1$), (4.1) can be rewritten as

$$(r^k, r^k) - (r^{k+1}, r^{k+1}) = C_j^i[R_j^i - S_j^i]$$

where

$$C_j^i = \frac{1}{1-\cos^2\theta_{ij}},$$

$$R_j^i = 2(r_i^k - r_j^k\cos\theta_{ij})(r^k, Aa^i) + 2(r_j^k - r_i^k\cos\theta_{ij})(r^k, Aa^j)$$

and

$$S_j^i = \frac{1}{1-\cos^2\theta_{ij}}[(r_i^k - r_j^k\cos\theta_{ij})^2(Aa^i, Aa^i) + 2(r_i^k - r_j^k\cos\theta_{ij})(r_j^k - r_i^k\cos\theta_{ij})(Aa^i, Aa^j) + (r_j^k - r_i^k\cos\theta_{ij})^2(Aa^j, Aa^j)].$$

To select a single step optimal method (i.e. one which would give a minimum value for $(r^{k+1}, r^{k+1}))i, j$ are chosen such that $C_j^i[R_j^i - S_j^i]$ is a maximum. We observe that $R_j^i - S_j^i \to 0$ as $r^k \to \emptyset$ ($\emptyset$ is the zero vector). In fact S_j^i will approach zero at a faster rate than R_j^i since $R_j^i = 0(r_i^k + r_j^k)$ and $S_j^i = 0(r_i^k + r_j^k)^2$. However, since $C_j^i \geq 1$, C_j^i dominates the product $C_j^i[R_j^i - S_j^i]$ as $r^k \to \emptyset$. Therefore a quasi-optimal selection criteria for the 2-dimensional x-projection method is to select pairs of rows of A in descending order of the values of C_j^i $(i, j = 1, 2, \ldots, n)$. All C_j^i $(i, j = 1, 2, \ldots, n)$ are computed initially and a stationary algorithm is obtained by selecting *a priori* pairs of rows of A based on the C_j^i values such that each row is used at least once and at most one row is used twice in each cycle. The largest C_j^i results from the two most parallel rows in A. Note in passing, for the 1-dimensional x-projection method that

$$(r^k, r^k) - (r^{k+1}, r^{k+1}) = r_i^2(Aa^i, Aa^i) + 2r_i(Aa^i, r^k)$$

which is not in a convenient form for determining in a like manner an *a priori* optimal sequence of 1-dimensional projections.

B. *Acceleration using geometric series*

Probably the most significant characteristic of the x-projection class of methods is that the ratio of the change vectors of the approximate solution vector as $r^k \to \emptyset$ converges to a constant. That is, at some cycle, m, the solution can be written

$$x = x^m + \Delta x^{m+1} + \Delta x^{m+2} + \cdots, \quad (\Delta x^{i+1} = x^{i+1} - x^i)$$

where $(\Delta x^{i+1}/\Delta x^i) = \rho$ for all $i > m$, $(\rho < 1)$ and $m, m+1, m+2, \ldots$ represents successive cycles of an x-projection method. Then the solution can be calculated directly:

$$x = x^m + \frac{\Delta x^{m+1}}{1-\rho}.$$

This result will not be proven. For some linear systems the iterative process converges before the mth cycle has been reached (i.e. before the value of ρ converges). The author has noticed this in general for linear systems where the r-projection and x-projection methods converge very rapidly such as diagonally dominant systems and unfortunately for some extremely ill-conditioned linear systems like Hilbert matrices. Thus the geometric acceleration may not always be able to be applied.

5. TEST PROBLEMS AND COMPARISONS

In this section six programs are compared. Each program is identified in the comparison tables by the following notation:

(1) G-S is the single step method of Gauss-Seidel.
(2) 1-X is the 1-dimensional x-projection method as described by Pizer[7].
(3) 1-XA is the 1-dimensional x-projection method where if possible a geometric acceleration is used.
(4) 2-X is the 2-dimensional x-projection method with quasi-optimal hyperplane selection as described in section 3.
(5) 2-XA is the 2-dimensional x-projection method with quasi-optimal hyperplane selection and if possible, geometric acceleration as described in section 3.
(6) 2-R is the 2-dimensional r-projection method with quasi-optional subspace selection as described in [11].

Each method was programmed in FORTRAN and executed from a library using a Honeywell Sigma 6. All calculations were performed in double precision and each test case used an initial starting vector of zero. Any I/O that was performed was not included in the C.P.U. time.

The order of operations (additions and multiplications) and the number of comparisons and library calls (absolute value, cosine, square root) for the overhead calculations (calculations that are performed only once) along with the calculations required at each cycle of iteration are depicted below for methods 1-X, 2-X (without hyperplane selection), 2-XA and 2-R.

	Overhead			Per cycle		
Method	Operations	Comparisons	Library	Operations	Comparisons	Library
1-X	$3n^2+2$	none	n	$4n^2+2n$	n	n
2-X	$4n^2+3n$	none	n	$4n^2+10n$	n	n
2-XA	$n^3+7n^2/2$	$(n^3-n^2)/2$	$n^2/2$	$4n^2+13n$	n	n
2-R	$n^3+n^2/2$	$(n^3-n^2)/2$	$n^2/2$	$2n^2$	n	n

The overhead and cycle calculations are of the same order for methods 1-X and 2-X. When hyperplane selection and geometric acceleration are added to the 2-X method producing method 2-XA, the overhead calculations jump from $0(4n^2)$ to $0(3n^3/2)$. However, the calculations per cycle remain the same order. Thus the advantages of hyperplane selection and geometric acceleration outweigh the additional calculations. This is verified in the following test cases.

A. *Special test case*

Test Case 1 is presented in Table 1 comparing the performance of the six programs. In Table 1 the coefficient matrix, angles between rows of the coefficient matrix, the constant vector, various pairings of row vectors and pairings of column vectors and final approximate solution vector are given. The column headings in Table 1 are described as follows:

Method—an abbreviation for one of the six programs.
Pair—indicates how the row vectors of the coefficient matrix are paired. A quaisi-optimal selection of projection hyperplanes is determined according to section 4 and labeled BEST (most parallel rows). In contrast the worst selection of projection hyperplanes is determined and labeled WORST (most orthogonal rows). Pairings associated with method 2-R are for columns vectors, of course.
Geo. Acc.—indicates if geometric acceleration to the solution could be performed (yes), could not be performed (no) or does not apply (—).
Error—inner product of the final residual vector. Powers of 10 are given in parenthesis. 1.9 (−8) denotes 1.9×10^{-8}.
Cycle—number of cycles required to obtain the solution.
Time—amount of C.P.U. time recorded in seconds. The Sigma 6 C.P.U. clock ticks occur every 500 ms. Each test case was run with little or nothing else in the system to minimize the number of swaps. In some instances test cases were rerun and the C.P.U. times were accurate within 1.5%.

In all cases the iterative process terminated when every corresponding component of two successive approximate solution vectors was within 0.000005.

Table 1. Results for Test Case 1

$$A = \begin{bmatrix} 100 & 245 & 630 & -75 & -896 & 0 & 0 \\ 147 & -25 & 300 & 756 & 0 & 13 & 0 \\ 0 & 123 & 452 & -1 & 753 & 249 & 0 \\ 12 & -568 & 30 & 21 & -78 & 0 & 85 \\ 0 & 0 & -23 & 563 & 752 & 843 & -1 \\ 145 & -20 & 1 & 7 & 30 & 6 & 0 \\ -1 & 0 & 43 & 8023 & 63 & 0 & 3 \end{bmatrix}, \quad x = \begin{bmatrix} 4.435283 \\ 3.141906 \\ -1.777156 \\ 0.412942 \\ -0.041611 \\ -0.190498 \\ -61.520275 \end{bmatrix}.$$

ANGLES =

Row						
2	81					
3	110	79				
4	94	85	102			
5	120	65	48	93		
6	95	77	80	82	80	
7	93	23	87	87	63	87
	1	2	3	4	5	6

(θ_j^i in degrees)

$$b' = (100, 350, -496, -7002, 143, 579, 3048)$$

2-XA BEST (7, 2), (5, 3), (6, 4), (5, 1)
2-XA WORST (7, 3), (4, 1), (6, 2), (5, 4)
2-R BEST (6, 5), (3, 1), (7, 2), (5, 4)
2-R WORST (4, 2), (7, 1), (6, 3), (7, 5)

Method	Pair	Geo. Acc.	Error	Cycle	Time
1. G-S	—	—	—	—	FAILS
2. 1-X	—	—	0.756(−7)	26,124	111.866
3. 1-XA	—	Yes	0.485(−14)	151	0.656
4. 2-X	BEST	—	0.765(−7)	6175	29.018
	WORST	—	0.812(−7)	25,383	121.018
5. 2-XA	BEST	Yes	0.274(−11)	28	0.174
	WORST	Yes	0.316(−7)	64	0.362
6. 2-R	BEST	—	0.353(−6)	137	0.404
	WORST	—	0.688(−3)	672	1.776

B. *Random coefficient matrix*

In test case 2 we tested the methods in the following manner. For a given dimension n, we generated a random matrix (a matrix consisting of random numbers between 0 and 1). We assumed a solution vector with components all 1's and generated an appropriate righthand side. The resulting systems were solved and the C.P.U. times and row or column vector pairings (BEST and WORST) were recorded. This procedure was repeated for each method for the same matrix. Results are displayed in Table 2. The iterative process terminated when every corresponding component of two successive approximate solution vectors was within 0.000005 (for dimensions 4, 8, 12, 16 and 20). For comparable accuracy a tolerance of 0.00001 was used for dimension 30, 40 and 50. G-S was not included in Table 2 because it failed in all cases.

Table 2. C.P.U. times in seconds for test case 2 (random matrices)

	Methods							
Dimension	1-X	1-XA	2-X BEST	2-X WORST	2-XA BEST	2-XA WORST	2-R BEST	2-R WORST
4	0.38	0.20	0.17	0.26	0.04	0.06	1.21	1.29
8	18.15	5.76	9.41	23.34	2.43	6.95	1.61	10.60
12	24.61	6.45	13.79	13.10	4.40	4.80	12.01	25.33
16	42.02	34.23	50.32	20.06	12.74	4.62	19.77	44.67
20	42.52	36.08	20.09	47.10	14.27	27.81	52.23	53.22
30	503.72	137.78	618.14	516.68	116.16	195.33	647.64	1052.90
40	710.97	712.93	492.80	766.59	294.41	511.79	714.41	1035.82
50	1018.16	1020.89	828.14	716.37	730.18	718.84	*	*

*Rate of convergence is extremely slow.

C. *Ill-conditioned matrices*

A classical example of ill-conditioned matrices is the set of *Hilbert matrices* H_n of order n with elements $H_n(i, j) = 1/(i + j - 1)$ (see [4]). In test case 3 we compare methods G-S, 1-X, 2-X, 1-XA, and 2-XA for various orders of Hilbert coefficient matrices. In each case a solution vector with components all 1's was assumed and the appropriate righthand side was generated. Table 3 gives the results of test case 3 where C.P.U. times for various dimensions of Hilbert matrices are given. In addition the maximum absolute difference of any component of the final approximate solution vector from 1 is given. This is used as a measure of accuracy rather than the inner product of the final residual vector. Results for 2-R are not included since it fell victim to round-off error propagation and failed to yield an accurate solution for $n > 8$. In fairness to method 2-R which theoretically guarantees to reduce the length of the residual vector and hence converge, we found the length of the residual vector was reported by the method as decreasing in cases when $n > 8$ but the approximate solution vector was diverging to infinity. In all cases a tolerance of 0.000005 was used to terminate the iterative process.

Table 3. Results for test case 3 (Hilbert matrices)

	Method									
	G-S		1-X		1-XA		2-X		2-XA	
Dimension	Max. Dif.	Time	Max. dif.	Time	Max. dif.	Time	Max. dif.	Time	Max. dif.	Time
4	0.0051	3.98	0.0296	13.99	0.0183	0.56	0.0139	0.24	0.0146	0.06
8	0.0197	19.30	0.0640	11.98	0.0637	0.92	0.0096	6.81	0.0092	0.56
12	0.0278	55.17	0.0664	122.31	0.0267	31.53	0.0191	5.71	0.0191	0.82
16	0.0259	100.95	0.0518	262.94	0.0403	34.82	0.0161	53.68	0.0067	14.25
20	0.0384	105.16	0.0558	314.60	0.0614	38.79	0.0128	90.23	0.0097	14.27
30	0.0280	492.97	0.0720	375.34	0.0720	377.21	0.0169	103.10	0.0186	16.87
40	0.0478	578.19	0.1542	450.17	0.0474	393.53	0.0267	103.79	0.0271	103.98
50	0.0214	1902.91	*	*	0.0867	1042.83	0.0149	794.24	0.0145	285.82

*Rate of convergence is extremely slow.

6. SUMMARY AND CONCLUSIONS

We feel the test cases provide fairly clear-cut conclusions. The 2-dimensional x-projection method with hyperplane selection and geometric acceleration, 2-XA proved to be the superior method over a wide range of test cases in both accuracy and C.P.U. time. 2-XA proved superior to the best known r-projection method, 2-R, and to the best known x-projection method to date, 1-X. G-S was included in the comparisons as a standard method for solving systems of linear equations to give a point of reference. 2-XA proved superior to G-S even when G-S did not fail (Table 3). The hyperplane selection criteria presented in section 4 proved in practice to be extremely successful for increasing the rate of convergence. Every test case showed this (2-X BEST vs 2-X WORST).

The quasi-optimal selection criteria proved successful in nearly every example. There were exceptions however, (Table 2, $n = 16$ and a few others) but for most cases it reduced the C.P.U. time significantly. The geometric acceleration technique presented in Section 4 proved extremely successful in practice. For the cases where it could be applied the C.P.U. time was reduced significantly and the accuracy improved by orders of magnitude. (See 2-XA BEST vs 2-X BEST in Table 1.) For the cases where it could not be applied the added cost in C.P.U. time to test for the geometric acceleration condition was insignificant. (See 2-XA BEST vs 2-X BEST in Table 3, $n = 40$; also 1-XA vs 1-X in Table 2, $n = 40$, 50 and in Table 3, $n = 30$.)

In conclusion, 2-XA like all projection methods is guaranteed to converge for arbitrary starting vector and nonsingular coefficient matrix. In addition, 2-XA has been shown to be competitive and in most cases superior to other well known iterative algorithms both in rate of convergence and accuracy. Furthermore, with ill-conditioned matrices 2-XA proved superior to the other methods tested. A program listing of method 2-XA in the form of a Fortran subroutine follows.

REFERENCES

1. L. Fox, *An Introduction to Numerical Linear Algebra.* Oxford University Press, New York (1964).
2. A. de la Garza, An Iterative Method for Solving Systems of Linear Equations. Union Carbide and Carbon Corp. K-25 Plant, Oak Ridge, Tennessee, Report K-731 (1951).

3. D. D. Georg and R. F. Keller, Criteria for Ordering Columns in Two-Dimensional Projection Methods, Ames Laboratory Report IS-3147, National Technical Information Service, Springfield, Virginia (1973).
4. R. T. Gregory and D. L. Karney, *A Collection of Matrices for Testing Computational Algorithms.* Wiley, New York (1969).
5. A. S. Householder, *The Theory of Matrices in Numerical Analysis.* Blaisdell, New York (1964).
6. R. F. Keller, A Single-Step Gradient Method for Solving Systems of Linear Equations, Technical Report 8, Mathematical Sciences, University of Missouri, Columbia, Missouri (1964).
7. S. M. Pizer, *Numerical Computing and Mathematical Analysis.* Science Research Associates, Chicago (1975).
8. H. D. Pyron, A Non-Stationary Optimizing of Projection Methods, Ph.D. Dissertation, Iowa State University, Ames, Iowa, Report IS-T-452, National Technical Information Service, Springfield, Virginia (1971).
9. M. Tokko and R. F. Keller, Optimal Three-Dimensional Projection Method for Solving Linear Algebraic Equations, Ames, Iowa Laboratory Report IS-3039, National Technical Information Service, Springfield, Virginia (1972).
10. R. L. Wainwright, Algorithms for Projection Methods for Solving Linear Systems of Equations, Ph.D. Dissertation, Iowa State University, Ames, Iowa, Report IS-3397, National Technical Information Service, Springfield, Virginia (1974).
11. R. L. Wainwright and R. F. Keller, Algorithms for Projection Methods for Solving Linear Systems of Equations. *Comput. Maths Appls* 3, 235–245 (1977).

```
      SUBROUTINE XPROJ2(N,A,B,X,TOL,IPRT,RDXDIF,ISUMY,MAXCYL)
      DOUBLEPRECISION A(N,N),B(N),X(N)
      DOUBLEPRECISION TOL,MAXDIF,DIFF,RDXDIF,DPA,DPR,DPSUM
      DOUBLEPRECISION DPSUM1,DPSUM2,COEF1,COEF2
      DOUBLEPRECISION DX(50),RDX(50),OLDX(50),OLDDX(50)
      DOUBLEPRECISION AA(50),PASTX(50)
      INTEGER ROW(50),ANGL(50,50),SP(51)

C
C*************************************************************
C**
C**   THIS SUBROUTINE SOLVES A NON-SINGULAR LINEAR SYSTEM
C**   OF EQUATIONS USING THE 2-DIMENSIONAL X-PROJECTION
C**   METHOD WITH QUASI-OPTIMAL HYPERPLANE SELECTION
C**   AND GEOMETRIC ACCELERATION.
C**
C**         ----- DESCRIPTION OF THE ARGUMENTS -----
C**
C**   N       - THE DIMENSION OF THE LINEAR SYSTEM.  THE
C**             SUBROUTINE IS SET UP FOR A MAX OF N=50
C**   A       - THE  COEFFICIENT MATRIX OF THE LINEAR SYSTEM
C**   B       - THE CONSTANT VECTOR OF THE LINEAR SYSTEM
C**   X       - THE APPROXIMATE SOLUTION VECTOR.  INITIALLY,
C**             X CONTAINS THE INITIAL GUESS VECTOR AND
C**             WILL CONTAIN THE SOLUTION UPON COMPLETION OF
C**             THE ALGORITHM. UNLESS SOME PRIOR
C**             KNOWLEDGE OF THE SOLUTION IS KNOWN THE ZERO
C**             VECTOR IS SUGGESTED FOR THE INITIAL GUESS
C**   TOL     - TOLERANCE LIMIT FOR DETERMINING CONVERGENCE. THE
C**             ITERATIVE PROCESS TERMINATES WHEN EVERY
C**             CORRESPONDING ELEMENT OF THE APPROXIMATE SOLUTION
C**             VECTOR FROM TWO SUCCESSIVE CYCLES IS WITHIN
C**             THIS VALUE. SUGGESTED VALUES RANGE FROM 0.00001
C**             TO 0.0000001.
C**   IPRT    - EVERY IPRT NUMBER OF CYCLES STATISTICS WILL BE
C**             GATHERED AND ANALYZED TO DETERMINE IF GEOMETRIC
C**             ACCELERATION CAN BE PERFORMED. IN MOST CASES
C**             VALUES FROM 25 TO 200 WORK QUITE WELL.
C**   RDXDIF- WHEN THE RATIO OF THE CHANGES IN THE APPROXIMATE
C**             SOLUTION VECTOR (AS GATHERED EVERY IPRT CYCLES)
C**             IS WITHIN THIS VALUE THEN GEOMETRIC ACCELERATION
C**             IS PERFORMED. **ONE MUST BE VERY CAREFUL WITH
C**             THIS ARGUMENT**  IF RDXDIF IS TOO LARGE PRE-
C**             MATURE ACCELERATION MAY OCCUR AND THE SYSTEM MAY
```

```
C**           DIVERGE.  OF COURSE, IF RDXDIF IS TOO SMALL
C**           ACCELERATION MAY NEVER OCCUR. RECOMMENDED VALUE
C**           IS .005.  ( ON THE HILBERT MATRIX OF SIZE>=40
C**           I USED .1 AND IT WORKED QUITE WELL. FOR A WELL-
C**           CONDITIONED SYSTEM .00005 MAY BE A BETTER VALUE.)
C**   ISUMY - EVERY ISUMY CYCLES PROGRESS OF THE SOLUTION IS
C**           REPORTED.  THE CYCLE NUMBER, APPROX. SOLUTION
C**           VECTOR AND INNER PRODUCT OF THE ERROR VECTOR
C**           ARE GIVEN.  IF ISUMY IS <= 0 THEN NO SUMMARY
C**           IS GIVEN.
C**   MAXCYL- THE MAXIMUM NUMBER OF CYCLES YOU WISH TO ITERATE.
C**           IF THE SOLUTION IS NOT DETERMINED BEFORE THIS,
C**           THE PROCESS IS HALTED WITH SUMMARY INFO. GIVEN.
C**
C*************************************************************
C ****** OVERHEAD CALCULATIONS ******************
C
C --- NORMALIZE THE COEF MATRIX, A ---------------
C
      DO 30 I=1,N
        DPSUM=0.
        DO 10 J=1,N
          DPA=A(I,J)
          DPSUM=DPSUM + DPA*DPA
10      CONTINUE
        DPSUM= DSQRT(DPSUM)
        DO 20 J=1,N
          A(I,J)= A(I,J)/DPSUM
20      CONTINUE
      B(I)=B(I)/DPSUM
30    CONTINUE
C
C  ---  END OF NORMALIZING THE A MATRIX -----------
C
C ----- CALCULATE THE ANGLES BETWEEN ROWS OF  A ---
C
      DPA=180.0
      DPA=DPA/3.14159
      DO 60 I=2,N
        L=I-1
        DO 50 J=1,L
          DPSUM=0.
          DO 40 K=1,N
            DPSUM=DPSUM+A(I,K)*A(J,K)
40        CONTINUE
          TEMP=DPSUM
          ANGL(I,J)=ACOS(TEMP)*DPA
C               MAKE ALL ANGLES <= 90 TO MORE EASILY
C               DETERMINE THE ROW VECTOR PAIRINGS
          IF(ANGL(I,J).GT.90) ANGL(I,J)=180-ANGL(I,J)
50      CONTINUE
60    CONTINUE
C
C
C --- END OF CALCULATING THE ANGLES ---------------
C
C --- SELECT QUASI-OPTIMAL ROW VECTOR PAIRS
C --- (MOST PARALLEL)
      ISP=1
      DO 70 I=1,N
        ROW(I)=0
70    CONTINUE
C
      DO 120 IDUMMY=1,999999
        MIN=300
        IROW=0
        ICOL=0
```

```
      DO 90 I=2,N
        IF(ROW(I).NE.0) GO TO 90
        L=I-1
        DO 80 J=1,L
          IF(ROW(J).NE.0) GO TO 80
          IF(ANGL(I,J).GE.MIN) GO TO 80
C             FOUND NEW MINIMUM
          MIN=ANGL(I,J)
          IROW=I
          ICOL=J
80      CONTINUE
90    CONTINUE
C
C
      IF(MIN.NE.300) GO TO 110
C
C ---DONE--FIX LOOSE ENDS IF N IS ODD---
C
      IF(N/2*2.EQ.N) GO TO 130
      DO 100 I=1,N
        IF(ROW(I).NE.0) GO TO 100
        SP(ISP)=I
        SP(ISP+1)=1
        IF(I.EQ.1) SP(ISP+1)=2
        GO TO 130
100   CONTINUE
      GO TO 130
C
C
110   SP(ISP)=IROW
      SP(ISP+1)=ICOL
      ISP=ISP+2
      ROW(IROW)=1
      ROW(ICOL)=1
120   CONTINUE
C
C
C --- END OF SELECTING THE ROW VECTOR PAIRS -------------
C
130   LASTSP = N
      IF( N/2*2.NE.N) LASTSP=N+1
C           LASTSP IN THE SIZE (LENGTH) OF SP
C --- CALCULATE THE INNER PRODUCTS (A(I),A(I+1)) --------
C --- CALCULATE 1 - ((A(I),A(I+1))**2)           --------
C --- I  AND  I1  ARE USED IN AA AND IN SP       --------
C --- K  AND  L  ARE USED IN A                   --------
C --- THE INNER PRODUCTS ARE CALCULATED FOR ONLY --------
C --- THE SELECTED PAIRS                         --------
C
      DO 150 I=1,LASTSP,2
        I1=I+1
        K=SP(I)
        L=SP(I1)
        DPSUM=0.
        DO 140 J=1,N
          DPSUM=DPSUM + A(K,J) * A(L,J)
140     CONTINUE
        AA(I) = DPSUM
        AA(I1) = 1. - DPSUM*DPSUM
150   CONTINUE
C
C --- END OF INNER PRODUCT CALCULATIONS -----------------
C
      DO 160 I=1,N
      OLDX(I)=X(I)
      OLDDX(I)=.00005
C         OLDDX IS INITIALIZED NON-ZERO TO PREVENT
C         ZERO DIVISION WHEN CALCULATING THE FIRST RDX
```

```
      PASTX(I)=X(I)
160   CONTINUE
C
C ****** END OF THE OVERHEAD (INITIALIZATION) CALCULATIONS ****
C ************************************************************
C
      IF(ISUMY.EQ.0) ISUMY=9999999
C ***** START OF THE CYCLE LOOP ***************************
C
      DO 280 ICYCLE=1,MAXCYL
C
C
C
C --- CHECK IF SUMMARY INFORMATION IS TO BE PRINTED -------
      IF(ICYCLE/ISUMY*ISUMY.NE.ICYCLE) GO TO 190
C --- PRINT SUMMARY INFORMATION
      DPR=0.
      DO 180 J=1,N
        DPSUM=0.
        DO 170 K=1,N
        DPSUM=DPSUM+ A(J,K)*X(K)
170     CONTINUE
        DPSUM=B(J)-DPSUM
        DPR=DPR+ DPSUM*DPSUM
180    CONTINUE
      WRITE(108,510) ICYCLE,DPR,(X(K),K=1,N)
C --- END OF PRINT SUMMARY INFORMATION --------------------
C
190   CONTINUE
C
C --- ITERATION LOOP.  COMPUTE THE NEW
C --- APPROX. SOLUTIION VECTOR BY PROJECTING
C --- ONTO X(K) AND X(L)
C
      DO 220 I=1,LASTSP,2
        I1=I+1
        K=SP(I)
        L=SP(I1)
C
C --- I AND I1 ARE USED IN AA AND IN SP
C --- K AND L ARE USED IN A
C
        DPSUM1=0.
        DPSUM2=0.
        DO 200 J=1,N
          DPSUM1=DPSUM1 + X(J)*A(K,J)
          DPSUM2=DPSUM2 + X(J)*A(L,J)
200     CONTINUE
        COEF1=((DPSUM2-B(L)) * AA(I)-DPSUM1+B(K)) / AA(I1)
        COEF2=((DPSUM1-B(K)) * AA(I)-DPSUM2+B(L))  / AA(I1)
C       COMPUTE NEXT APPROX. SOLUTION VECTOR
        DO 210 J=1,N
          X(J)=X(J) + COEF1*A(K,J)+COEF2*A(L,J)
210     CONTINUE
220   CONTINUE
C
      MAXDIF=0.
      DO 230 J=1,N
        DIFF=DABS(X(J)-PASTX(J))
        IF(DIFF.GT.MAXDIF) MAXDIF=DIFF
        PASTX(J)=X(J)
230   CONTINUE
      IF(MAXDIF.LE.TOL) GO TO 290
C
C
C
C --- END OF THE CYCLE CALCULATIONS -------------------
```

```
      IF(ICYCLE/IPRT*IPRT.NE.ICYCLE)GO TO 280
C
C
C --- ATTEMPT TO ACCELERATE
      DO 240 J=1,N
        DX(J)=X(J)-OLDX(J)
        OLDX(J)=X(J)
        RDX(J)=DX(J)/OLDDX(J)
        OLDDX(J)=DX(J)
240   CONTINUE
C
C --- IF RDX VALUES ARE ALL WITHIN RDXDIF OF EACH
C --- OTHER THEN ACCELERATE TO THE SOLUTION
C
      L=N-1
      DO 260 I=1,L
        K=I+1
        DO 250 J=K,N
          IF(DABS(RDX(I)-RDX(J)).GT.RDXDIF)  GO TO 280
250     CONTINUE
260    CONTINUE
C
      DO 270 I=1,N
        X(I)= (X(I)-DX(I)) + DX(I)/(1.0-RDX(I))
        PASTX(I)=X(I)
        OLDDX(I)=0.000001
C          OLDDX IS MADE NON-ZERO TO PREVENT ZERO
C          DIVISION ON THE NEXT CYCLE
        OLDX(I)=X(I)
270   CONTINUE
C
C
280   CONTINUE
C
C ************ END OF THE CYCLE ****************************
C
C
C ************ SOLUTION FOUND ******************************
C
290   CONTINUE
C     CALCULATE (R,R)
      DPR=0.
      DO 310 J=1,N
        DPSUM=0.
        DO 300 K=1,N
        DPSUM=DPSUM + A(J,K)*X(K)
300     CONTINUE
        DPSUM=B(J)-DPSUM
        DPR=DPR + DPSUM*DPSUM
310   CONTINUE
C --- PRINT THE SOLUTION
      WRITE(108,500) ICYCLE,DPR,(X(K),K=1,N)
C
500   FORMAT('0'//,'SOLUTION'/'NO OF CYCLES = ',
     X  I9,'  (R,R) = ',E18.9/'0SOLUTION VECTOR FOLLOWS',
     X /(' ',8(F13.7,1X)))
510    FORMAT(' AT CYCLE = ',I9,'  (R,R) = ',E18.9/
     X ' APPROX. SOLN. VECTOR FOLLOWS'/
     X (' ',8(F13.7,1X)))
       RETURN
       END
```